TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

GRAFICKÝ EDITOR ČÍSLICOVÝCH OBVODŮ V HTML5

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Jaroslav Řehák**
Vedoucí práce: Ing. Martin Rozkovec, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

HTML5 DIGITAL ELECTRONICS EDITOR

Diploma thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Jaroslav Řehák**
Supervisor: Ing. Martin Rozkovec, Ph.D.

Liberec 2014

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu diplomové práce, panu Ing. Martinu Rozkovcovi, Ph.D., za jeho cenné rady a pomoc při zpracovávání. Rovněž bych rád poděkoval své přítelkyni Bc. Petře Prušinovské za její neustálou podporu.

Abstrakt

Diplomová práce se věnuje grafickému editoru číslicových obvodů v HTML 5. Práce je rozdělena na dva celky. V prvním celku nalezneme seznámení s danou problematikou a výčet použitých technologií. Druhý celek podrobně vysvětluje zacházení s naprogramovanou aplikací a možnosti propojení s jinými aplikacemi. Jsou zde popsána všechna hradla, se kterými lze pracovat. Výstupem aplikace jsou dva soubory. Jedním je testovací soubor pro simulaci obvodu a druhým je samotné navržené schéma, popsané jazykem VHDL. Do aplikace lze nahrát již hotové schéma přetažením souboru VHDL na určené místo. V rámci druhého celku jsou ještě popsány struktury všech souborů, s nimiž se pracuje. Výsledkem práce je také knihovna prvků, kterou je možné, dle návodu v této zprávě, dále rozšiřovat.

Klíčová slova

HTML 5, VHDL, číslicové obvody, diplomová práce, grafický editor

Abstract

Diploma thesis deals with HTML 5 digital electronics editor. Work is divided into two main parts. In the first part there is explained problematic of a digital electronics and all the used technologies are introduced. The second part explains controlling of the application in details and possibilities of connection with other applications. There is also description of all gates and circuits which can be used. Outputs of the application are two files. The first is a testing file for the circuit simulation and the second is designed circuit itself described using VHDL. There is a function for uploading files with designed circuits by so called „drag and drop“ function into a prepared space in the page. In the second part there are also described the structures of all files which one can work with. The result of the work is also the library of gates, which can be expanded through the manual described in this thesis.

Keywords

HTML 5, VHDL, electronic circuits, diploma thesis, graphical editor

Obsah

Seznam tabulek

Seznam zdrojových kódů

Seznam ilustrací

1 Úvod

Cílem diplomové práce je naprogramování webového grafického editoru číslicových obvodů v HTML 5 [17] a současně vytvoření knihovny elektronických číslicových prvků. Knihovna by měla být dodatečně rozšiřitelná o další číslicové prvky. Výstupem editoru bude soubor, v němž by mělo být textově popsáno uživatelem vytvořené schéma, které má strukturou odpovídat popisu jazyka VHDL [12]. Rovněž by webová aplikace měla načítat a zobrazovat stejně formátované soubory.

Zabývám se problematikou programování webových stránek, kaskádových stylů a JavaScriptu, a proto pro mě byl velkou motivací fakt, že neexistuje žádné řešení v podobě malého nástroje pro návrh jednoduchých obvodů. Navíc je práce zamýšlena jako podpora výuky zde na Technické univerzitě v Liberci. Využití by mohla nalézt zejména v předmětech, které jsou svázány s číslicovou technikou. Jelikož má být aplikace webová, každý uživatel by mohl zadané úkoly z číslicových předmětů vypracovávat odkudkoliv, kde je k dispozici internetové připojení. Aplikace bude volně dostupná nejen pro studenty, ale i pro veřejnost.

V první, teoretické části, se budu zabývat rešerší možných řešení a postupů při tvorbě webových aplikací. Dále čtenáře seznámím se standardem HTML 5 a jeho hlavními prvky, mezi nimiž se nejvíce zaměřím na prvek plátno. Představím též použité programy, jež mi pomohly při tvorbě práce, a současně i potřebné programové knihovny. V této části objasním i základy strukturálního popisu obvodů v jazyce VHDL.

V druhé, praktické části, budu popisovat postup tvorby výsledné webové aplikace, její ovládání a způsob použití knihoven. Rozeberu strukturu výstupního VHDL souboru a cestu, jakou bude možné rozšiřovat databázi prvků. Další věcí, kterou popíši, je možnost propojení se standardními simulátory VHDL. V závěru nabídnu možnosti rozšíření této práce.

2 Teoretická část

Programování zadané aplikace předcházelo prostudování mnoha odborných článků a internetových stránek zabývajících se tématy úzce spjatými s mou prací. Například na webu StratoSim [14] je vytvořen nástroj, jenž danou problematiku řeší obdobným způsobem. Na celé ploše uvnitř prohlížeče je rozprostřen prvek plátno, na který se přidávají různé členy. Projekt obsahuje ovšem pouze základní logické členy (AND, NOR, XOR,…) a je zaměřen především na RC obvody, jež se skládají z rezistorů a kondenzátorů. Výstupem navrženého schématu může být pouze pdf soubor. Mimo jiné zde musí být každý uživatel zaregistrován pomocí emailové adresy. Druhým podobným projektem je Wronex [18]. Zde je již větší nabídka logických prvků pro navrhování schémat (NOT, OR, MUX, JK latch,…). Navržené schéma lze uložit pouze do textového souboru nebo do lokálního úložiště pro pozdější načtení. Nenalezl jsem však žádný projekt, který by vyhovoval mým požadavkům. Jediné, co se podobá tématu, je JavaScriptová knihovna JointJS [3], jíž jsem se nechal inspirovat. O této knihovně se zmíním v kapitole 2.2.

Následně jsem prozkoumal dostupné knihovny, které by mi mohly pomoci v tvorbě 2D grafiky. Vybíral jsem z několika volně stažitelných, a to proto, abych neporušoval licence či práva. Již z počátku jsem některé z nich vyřadil, protože neobsahovaly prvky důležité pro mou práci nebo byly příliš rozsáhlé a zbytečně náročné na fyzické zdroje počítače. Knihovny, jež jsem pro svou práci vybral, budou podrobněji popsány.

2.1 Jazyk VHDL

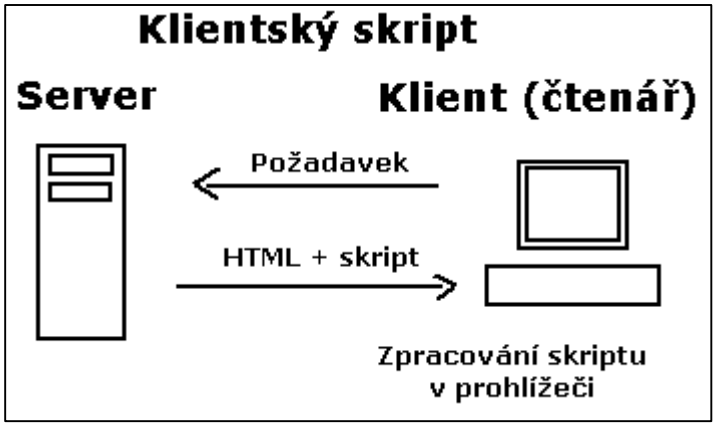
Dalším krokem bylo prostudování knihy zabývající se VHDL [12], z níž jsem získal základní znalosti o tomto popisovacím jazyku. Mimo jiné jsou zde vysvětleny veškeré základní pojmy důležité pro pochopení syntaxe, způsobu zápisu apod. Díky této publikaci jsem se dozvěděl, jaké struktury jsou platné pro zápis a čtení VHDL souborů a rovněž to, jakým způsobem skládat jednotlivé komponenty vyskytující se v této práci.

VHDL je jazyk vysoké úrovně, pomocí kterého se popisuje hardware. Slouží k navrhování a simulaci integrovaných a analogových obvodů. Základní popis hardware je tvořen instancemi komponent. Každá komponenta obsahuje tzv. entitu, v níž se nachází definice rozhraní komponenty, což jsou vstupy a výstupy. Druhou součástí

komponenty je tzv. architektura, v níž se udává funkce a chování. Tato obsahuje deklarační a příkazovou část. Tímto způsobem definovanou strukturu uplatním ve mnou vytvořené knihovně prvků. Struktura přijde na řadu při vytváření výstupního VHDL souboru. Prvky použité v navrženém schématu se načtou a přiřadí se jim jejich vnitřní struktura. Dle návrhu se doplní zbylý kód a vše se uloží do výstupního souboru. Ten bude dále možné použít k simulaci v jakémkoliv externím programu, který tyto druhy souborů umí zpracovat.

2.2 JavaScript

Největší část práce je založena právě na JavaScriptu [20] a využívá knihovny, jež jsou v něm rovněž napsány. Jedná se o skriptovací jazyk, schopný pracovat s DOM objekty standardu HTML. Dokáže k nim přistupovat pomocí různých příkazů, číst a měnit jejich obsah. Dále může reagovat na události, které nastanou například při kliknutí na tlačítko, přejetí přes textové pole, při odeslání různých formulářů a podobně.



Obrázek 1: Popis architektury server - klient

„JavaScript je klientský skript (viz Obrázek 1). To znamená, že se program odesílá se stránkou do prohlížeče a teprve tam je vykonáván. Protikladem klientských skriptů jsou skripty serverové, které jsou vykonávány na serveru a na klienta jdou už jen výsledky.“ [7]

2.2.1 JavaScriptové knihovny

Pro splnění zadání této práce by byla jedna z možností naprogramování vlastního rozhraní a vlastních knihoven, s nimiž bych poté pracoval. Po delším zkoumání a přečtení článků [2] [3] [8] zabývajících se programováním webových aplikací, jsem

se však dočetl o několika již naprogramovaných JavaScriptových knihovnách, které by se daly v diplomové práci využít.

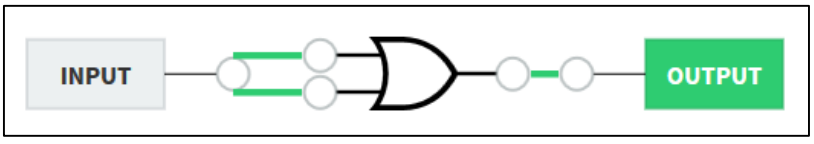
První knihovnou je jQuery [8], jež zjednodušuje psaní JavaScriptového kódu (viz Obrázek 2). Ta je důležitá i pro ostatní knihovny pomocí jQuery napsané.

```
$(document).ready(function(){
        $("p").click(function(){
                $(this).hide();
        });
});
```

Obrázek 2: Ukázka funkce z jQuery

Je založena skupinou nazvanou The jQuery Foundation a je šířena pod licencí MIT dovolující volné šíření za podmínek, že se ponechá nedotčená hlavička s autorstvím. Jednou z výhod je minimalizace množství kódu, což také zvýší jeho přehlednost. Je multiplatformní, díky čemuž ji můžeme využít ve všech operačních systémech. Její kladnou vlastností je funkčnost ve všech prohlížečích bez rozdílu. Základem je znak „**$**“ přítomný na začátku každého příkazu. Dále se používají například selektory identifikátorů, začínající znakem „**#**“ nebo tříd, kde je použitým znakem tečka. Dalšími prvky jsou práce s animacemi, metody filtrování, různé efekty a zachytávání událostí. To vše s nepoměrně kratším kódem, než kdyby to bylo napsáno čistým JavaScriptem.

Další knihovnou, umožňující vizualizovat a pracovat s grafy a diagramy ve 2D vektorové grafice, je JointJS [3]. Podléhá MPL verze 2.0. Obsahuje základní elementy, jimiž jsou obdélník, kruh, elipsa, text, obrázek a cesta. Všechny prvky jsou založeny na vektorové grafice, tudíž jsou vykreslovány za chodu. Knihovna též podporuje serializaci a deserializaci pro formát JSON. Nechybí zde ani podpora pro dotyková zařízení, možnost přibližování a oddalování, nastavitelná propojení prvků, různé filtry, gradienty, animace a hierarchické diagramy. Ke knihovně existují i další rozšíření. Jedním z několika, kterým jsem se nechal inspirovat, je rozšíření Logic Circuits, což jsou v překladu do českého jazyka logické obvody. Toto rozšíření má nadefinováno několik základních prvků: hradlo, vstupní porty, výstupní porty, obrázek hradla a vodič.



Obrázek 3: Náhled na použitou knihovnu Logic Circuits

Rozšiřuje základní knihovnu o funkci mající schopnost za chodu simulovat právě navrhovaný obvod (viz Obrázek 3). Na pozadí probíhá kontrola všech prvků. Aktivní vodič, jímž prochází logická jedna, je kaskádovým stylem obarven zelenou barvou. Proti tomu neaktivní vodič, kde je logická nula, je obarven šedou barvou. Rozšíření však obsahuje pouze několik základních hradel: REPEATER, NOT, OR, XOR, AND, NAND, NOR, XNOR, VSTUP a VÝSTUP.

Poslední knihovnou je Raphaël [2]. Měla by zjednodušovat práci s vektorovou grafikou. U knihovny je využita rovněž MIT licence, která dovoluje její bezplatné využívání. Kromě zjednodušení práce s vektorovou grafikou lze využít rovněž metody pro animace různých objektů, zachytávání a spouštění událostí na stránce. Nechybí zde podpora pro dotyková zařízení a maticové operace. U knihovny jsem se bohužel setkal s množstvím problémů, a proto jsem ji vyřadil již v rané fázi programování. Měla příliš složitou strukturu a navíc zde nebyla přímá podpora ani žádné rozšíření ulehčující práci s logickými prvky.

2.3 Webový standard HTML 5

V současné době je poslední schválenou specifikací HTML verze 4.01. Hlavními nevýhodami je, že pro přehrávání jakéhokoliv multimediálního obsahu nebo pro zobrazení animací, či pro zobrazení grafiky, je nutné do prohlížečů doinstalovat další doplňky. Mezi používaná rozšíření pro prohlížeče patří dvě nejznámější, a to Flash player od firmy Adobe, pro něhož se programuje objektově orientovaným jazykem ActionScript, a jazyk Java, který využívá pro zobrazování grafiky jazyk JavaFX. Aby uživatelé nemuseli instalovat žádné další doplňky, zvolil jsem novou verzi standardu, jímž je HTML 5. Jedná se o nový standard pro internetové stránky, zatím stále ve fázi testování. Schválení je plánováno na třetí čtvrtletí roku 2014. Mezi novými prvky této verze jsou:

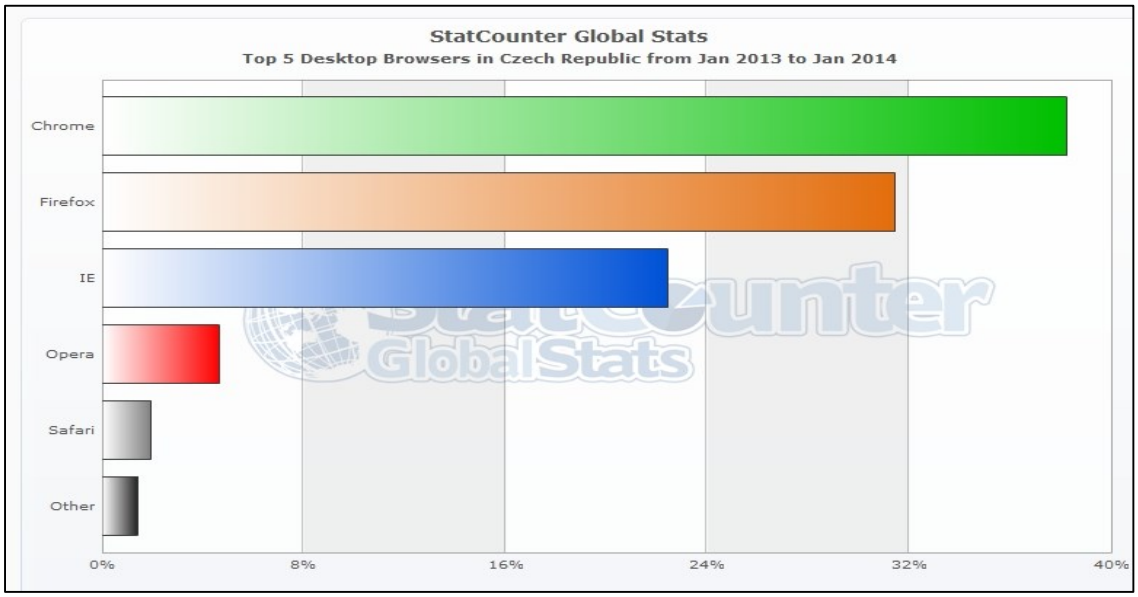
a) Plátno (canvas) – prostředek pro vykreslování 2D a 3D grafiky
b) Elementy video a audio – pro snadnou práci s multimédii
c) Místní úložiště (local storage) – rychlejší a bezpečnější ukládání dat
d) Prvky rozdělení stránky – záhlaví, zápatí, článek, navigace, sekce
e) Prvky formuláře – kalendář, datum, čas, email, …

f) „Táhni a pusť“ (drag and drop) – Možnost přetahování daných prvků na stránce

Jedním z hlavních důvodů pro vznik standardu HTML 5 bylo vykreslování grafiky přímo v prohlížeči, bez potřeby instalace dalších rozšiřujících doplňků, mezi nimiž je například právě přehrávač Flash player. 2D i 3D grafika se na internetové stránce vykresluje do výše zmíněného plátna tvořeného elementem „**<canvas>**.“ Samotné kreslení je prováděno pomocí skriptovacího jazyku JavaScript. Pro 3D grafiku existuje aplikační rozhraní WebGL napsané v JavaScriptu. Od rozšíření HTML 5 mezi uživatele uplynula již delší doba, a proto má v současnosti solidní podporu všech nejvíce používaných prohlížečů. Výhodou je multiplatformnost, což znamená, že jej můžeme využít v jakémkoliv zobrazovacím zařízení, a to od mobilního telefonu až po chytré televize s internetovým prohlížečem. Má také podporu pro kaskádové styly ve verzi číslo 3.

2.3.1 Internetové prohlížeče

Průběh a výsledky práce jsem testoval ve čtyřech, dle mého názoru nejpoužívanějších prohlížečích, jimiž jsou Mozilla Firefox [10], Google Chrome [5], Internet Explorer [9] a Opera [11]. Všechny byly v jejich nejnovějších dostupných verzích. V České republice patří za poslední rok k nejpoužívanějším prohlížečům Chrome. Dle statistik dosahuje necelých čtyřiceti procent uživatelů (viz Obrázek 4).



Obrázek 4: Statistický přehled nejpoužívanějších internetových prohlížečů

Navrhovaná webová aplikace v HTML 5, využívající JavaScript, by měla být použitelná ve všech prohlížečích, nejen v těch nejpoužívanějších. Je však nutné dodat, že aplikace bude fungovat pouze v těch prohlížečích, které podporují všechny prvky HTML 5 využité v této práci. To znamená, že pokud nějaký starší prohlížeč nebude podporovat prvek plátno, potom aplikace nebude fungovat.

2.4 Aplikace XAMPP

Pro práci na již zmiňované webové aplikaci jsem použil bezplatnou aplikaci s názvem „**XAMPP**“ podléhající GNU GPL, tudíž je volně využitelná i pro komerční použití. Program na počítači vytvoří virtuální webový server, pomocí něhož mohu testovat funkčnost napsaného kódu. Jedná se ve své podstatě o balíček vývojových nástrojů, mezi něž patří server „**Apache**,“ databázový systém „**MySQL**,“ podpora programování v jazyce PHP a skriptování v jazyce Perl. V práci si však vystačím pouze s „**Apache**“ serverem. Po spuštění libovolného internetového prohlížeče se navolí adresa místní sítě, což je buď příkaz „**localhost**“ nebo adresa „**127.0.0.1.**“ Poté se již načte hlavní stránka webové aplikace. Program „**XAMPP**“ načítá internetové stránky z podsložky nazvané „**htdocs**.“ Ta je umístěna v kořenovém adresáři nainstalované aplikace.

2.5 Kaskádové styly

V práci jsem rovněž použil kaskádové styly [16], jež upravují grafický vzhled internetových stránek. Jedná se o textový soubor, do něhož se pomocí tzv. tagů zapisují jednotlivé části webové stránky, jimž se poté přiřadí různé vlastnosti. Kaskádové styly vznikly za účelem oddělení vzhledu a textu. Díky tomu je možné pro stejný text vytvořit mnoho grafických variací, mezi kterými poté můžeme přepínat. Je také možné nastavovat podmínky, podle nichž se rozhodne, jaký styl bude použit. Toho se využívá například pro zobrazení jiného stylu na mobilním telefonu, kde navržené prostředí musí korespondovat s velikostí displeje. Aktuálně jsou kaskádové styly ve verzi 3.

3 Praktická část

Tato část práce je zaměřena na postup při tvorbě aplikace a též na využité knihovny. Bude zde detailně popsáno ovládání aplikace a struktury souborů, které jsou důležité pro pochopení a případné rozšíření. Poskytnu i podrobný návod na přidávání nových prvků a vysvětlím možnosti, jakými lze vyexportované soubory dále využít, v jakých programech je možné tyto soubory simulovat a zda existují nějaká online řešení.

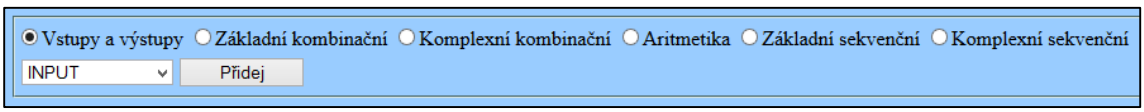
3.1 Postup tvorby webové aplikace

Základním stavebním prvkem, jenž jsem vytvořil jako první, byla webová stránka, na níž jsem postupně rozmístil jednotlivé komponenty. Skládá se ze souboru „**index.html**“ obsahujícího hlavičku, tělo a zápatí. V hlavičce je nadefinována znaková sada UTF-8. Je zde vložen i odkaz na kaskádový styl, kterým je web nastylován. V těle webové stránky jsou deklarace přepínačů, a to v takovém množství, kolik je nadefinovaných skupin prvků, jež bude možné vložit do schématu. Následuje několik tlačítek, o nichž se zmíním v příští kapitole. Nalezneme zde rovněž několik nadpisů, pod nimiž je nadefinováno samotné plátno. Ve zdrojovém kódu je reprezentováno pouze jako blok, který má nastaven vlastní identifikátor. Nadefinování plátna je poté provedeno až pomocí JavaScriptu. Na konci kódu jsou deklarovány všechny potřebné skripty, mezi něž patří tyto: „**fileLoader.js**, **jquery.js**, **joint.js**, **joint.shapes.mylib.js, main.js**.“

3.1.1 Ovládání aplikace

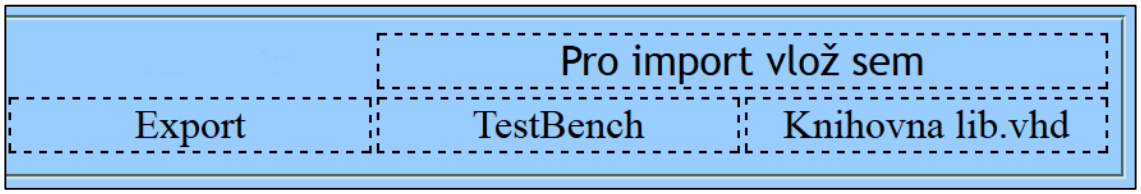
Hlavní stránka aplikace se skládá z několika částí, které následně popíši směrem shora dolů.

a) Horní panel



Obrázek 5: Náhled na panel aplikace

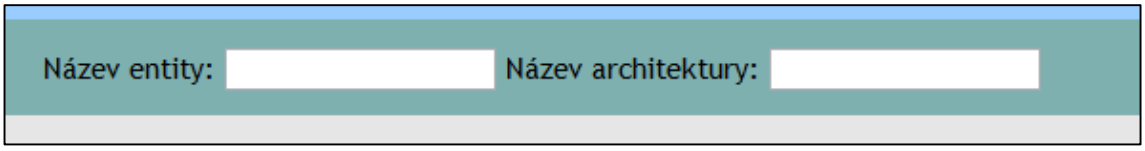
Nahoře zleva se nachází panel s výběrem skupin prvků. Vždy lze vybrat pouze jednu skupinu (viz Obrázek 5). Těsně pod tímto panelem je rozevírací seznam, do něhož se načtou prvky z vybrané skupiny. Vedle rozevíracího seznamu se nachází tlačítko „**přidej**," které přidá do plátna vybraný prvek.



Obrázek 6: Tlačítka pro generování souborů, nahrávání a stažení knihovny

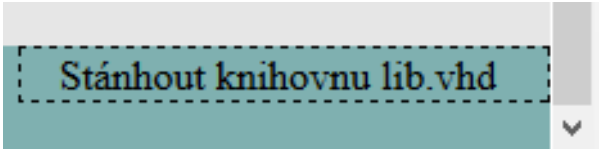
V rámci panelu jsou vpravo od skupin prvků umístěna dvě funkční tlačítka (viz Obrázek 6). Jedním z nich je „**TestBench**," jež vygeneruje soubor vyjadřující testovací prostředí pro kontrolu správnosti navrhovaného schématu. Druhým tlačítkem je „**Export**," které vygeneruje požadovaný strukturovaný soubor s navrženým schématem v jazyce VHDL. V rámci horního panelu je ještě vpravo nahoře obdélníková oblast, kam lze přetáhnout z disku soubor, jímž chceme importovat do plátna uložené schéma obvodu. Zde je dle obrázku text „**Pro import vlož sem**."

b) Střední panel



Obrázek 7: Textová pole pro název entity a architektury

Ve středním panelu se nacházejí dvě textová pole, do nichž je možné zadat název entity vytvářeného obvodu a také název architektury (viz Obrázek 7). V případě, že tato pole zůstanou nevyplněna, jsou předdefinovány výchozí hodnoty, aby nedošlo k neočekávanému chování aplikace. Vpravo od těchto polí je velkým písmem název webové aplikace. V blízkosti pod těmito prvky leží prvek plátno, do kterého se vykresluje veškerá grafika. Šířka plátna má necelých sto procent šířky horizontálního rozlišení monitoru a výška plátna je jeden a půl násobku vertikálního rozlišení. Tyto hodnoty jsou optimální pro dostatečný prostor k navrhování obvodů.



Obrázek 8: Tlačítko pro stažení knihovny

V pravém dolním rohu webové stránky byl umístěn odkaz na stažení knihovny „**lib.vhd**,**“** která je nezbytná pro překlad vygenerovaného souboru v simulátoru VHDL. Pro lepší přehlednost byl odkaz přesunut do horního panelu (viz Obrázek 6 a 8).

3.1.2 Použití knihoven a skriptů

Skript fileLoader.js obsahuje funkční kód, jenž obstarává načtení souboru pomocí metody „**XMLHttpRequest**“ a dále získává text z daného souboru. Další skript „**jquery.js**“ je kompletní knihovna pro ušetření práce s JavaScriptem. „**Joint.js**“ je knihovna, popsaná v kapitole 2.2.1, k níž je přiřazen její vlastní kaskádový styl, udávající vzhled prvků, které jsou touto knihovnou vytvářeny. Využívám ji k deklaraci plátna a také jako zdroj, z něhož čerpá příkazy mnou vytvořená knihovna joint.shapes.mylib.js. Její struktura je popsána v kapitole 3.4. Posledním skriptem je „**main.js**,“ který obsahuje kompletní seznam funkcí, jimiž obsluhuji webovou stránku. Zdrojový kód je okomentován a přehledně rozdělen do částí spolu úzce souvisejících.

3.2 Popis logických prvků dle skupin

V následujících podkapitolách jsou rozebrány skupiny logických členů. Celkem je zde šest skupin, přičemž každá z nich obsahuje rozdílné množství prvků dané kategorie. U každé skupiny je tabulka obsahující výčet všech členů. Každý člen je popsán názvem, všemi vstupy, výstupy, popiskem a poznámkou, v níž jsou vysvětleny některé vstupy a výstupy. Do všech skupin je rovněž vložen obrázek s několika zástupci z dané skupiny.

3.2.1 Vstupy a výstupy

a) INPUT

Prvek „**INPUT**“ představuje definici vstupu. Nemá žádný vstupní port, ale pouze jeden výstupní port označený písmenem „**q**.“ Popisek prvku je složen z písmena „**X**“ a čísla určujícího pořadí přidání do plátna. Pořadí se stále přičítá, dokud nedojde k obnovení celé webové aplikace. Z výstupního portu je možné množit vodič, protože jej nelze

rozvětvovat. V případě potřeby rozvětvení vodiče je nutné táhnout z příslušného výstupního portu vodič další.

b) OUTPUT

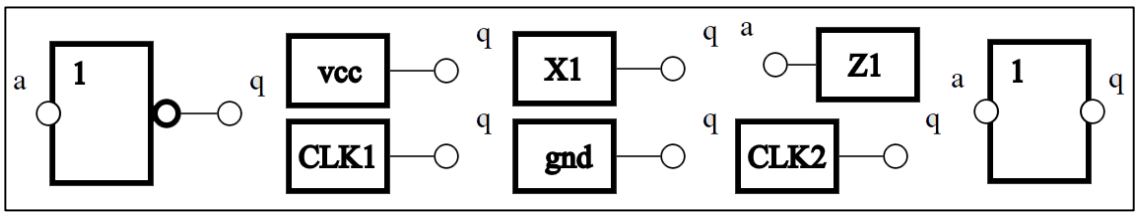
„**OUTPUT**“ zastupuje naopak výstup. Nemá tudíž žádný výstupní port, ale pouze jeden vstupní port. Do vstupního portu lze vést vždy pouze jeden vodič. Popisek výstupu je označen písmenem „**Z**“ a také příslušným pořadím, v jakém byl prvek přidán do plátna. Před každý výstup je nutné vložit prvek „**BUF**,“ který ošetřuje redundanci, jež by vznikla v případě, kdy by byl výstup libovolného hradla znovu použit a zároveň zapojen na výstup. Toto nelze dopustit, a proto je takovéto opatření nezbytné. Pro zjednodušení je zároveň s výstupem vložen zmíněný prvek „**BUF**.“

c) CLK

Hradlo „**CLK**“ popisujeme jako hodinový signál. Popisek obsahuje číslování. Opět je zde pouze jeden výstupní port a žádný vstupní. Výstup je označen písmenem „**q**.“ Hodinový signál má návaznost na generování výstupních souborů. Pokud je tento prvek přítomný na plátně, potom se navíc do souboru vloží příslušný kód, který je zmíněn ve struktuře výstupních souborů.

Tabulka 1: Seznam členů v kategorii vstupy a výstupy

Název	Vstupy	Výstupy	Funkce	Poznámka
INPUT	-	q	Vstup	
OUTPUT	a	-	Výstup	Vždy vstupem BUF
VCC	-	q	Napěťový vodič	
GND	-	q	Zemnící vodič	
CLK	-	q	Hodinový signál	
BUF	a	q	Opakovač	
INV	a	q	Invertor	



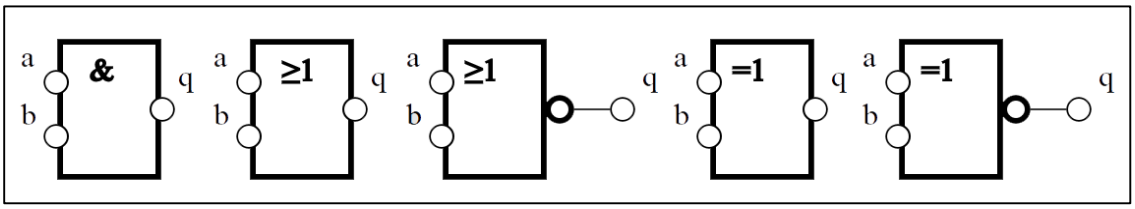
Obrázek 9: Vstupy a výstupy

3.2.2 Základní kombinační členy

V následující tabulce je seznam všech základních kombinačních členů, které je možné přidat do plátna ve webové aplikaci.

Tabulka 2: Seznam základních kombinačních členů

Název	Vstupy	Výstupy	Funkce
AND	a, b	q	Logický součin
AND3	a, b, c	q	Logický součin
AND4	a, b, c, d	q	Logický součin
NAND	a, b	q	Logický negovaný součin
NAND3	a, b, c	q	Logický negovaný součin
NAND4	a, b, c, d	q	Logický negovaný součin
NOR	a, b	q	Logický negovaný součet
NOR3	a, b, c	q	Logický negovaný součet
NOR4	a, b, c, d	q	Logický negovaný součet
OR	a, b	q	Logický součet
XOR	a, b	q	Exkluzivní logický součet
XNOR	a, b	q	Exkluzivní negovaný součet



Obrázek 10: Výběr hradel ze základních kombinačních členů

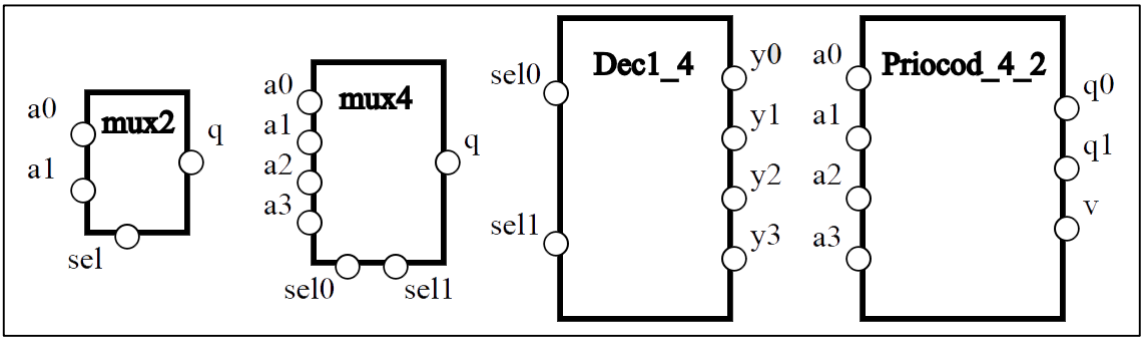
3.2.3 Komplexní kombinační členy

Podkapitola popisuje komplexní kombinační členy včetně obrázku některých zástupců.

Tabulka 3: Seznam komplexních kombinačních členů

Název	Vstupy	Výstupy	Popis	Poznámka
MUX2	a0, a1, sel	q	Multiplexor	2 vstupy
MUX4	a0, a1, a2, a3, sel0, sel1	q	Multiplexor	4 vstupy
MUX8	a0, a1, a2, a3, a4, a5, a6, a7, sel0, sel1, sel2	q	Multiplexor	8 vstupů
DEC14	sel0, sel1	y0, y1, y2, y3	Dekodér	4 výstupy

DEC18	sel0, sel1, sel2	y0, y1, y2, y3, y4, y5, y6, y7	Dekodér	8 výstupů
PRIOCOD42	a0, a1, a2, a3	q0, q1, v	Prioritní kodér	4 vstupy / 2 výstupy
PRIOCOD83	a0, a1, a2, a3, a4, a5, a6, a7	q0, q1, q2, v	Prioritní kodér	8 vstupů / 3 výstupy



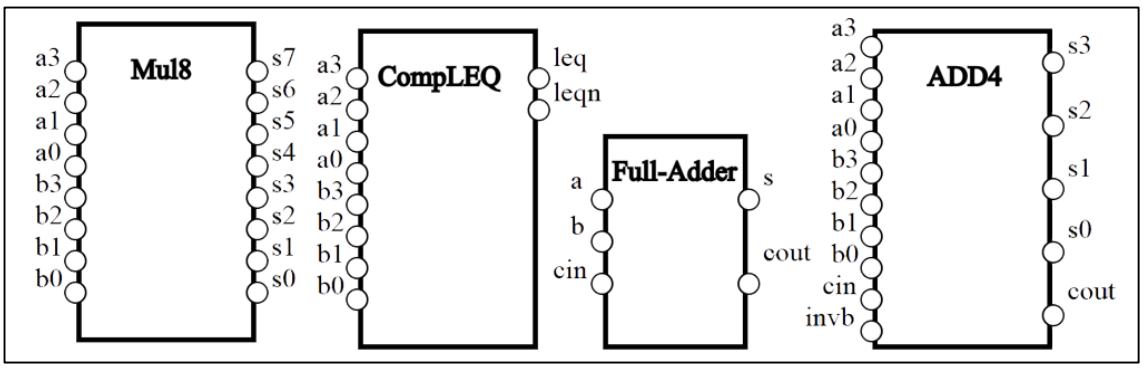
Obrázek 11: Výběr komplexních kombinačních členů

3.2.4 Aritmetické členy

Mezi aritmetické členy jsem zařadil níže zmíněné členy, a to z důvodu, že provádějí aritmetické operace sčítání, násobení či porovnávání.

Tabulka 4: Seznam aritmetických členů

Název	Vstupy	Výstupy	Popis	Poznámka
HALFADDER	a, b	s, c	Jednobitová sčítačka	c – přenosový bit
FULLADDER	a, b, cin	s, cout	Úplná sčítačka	cin, cout – přenosové bity
ADD4	a0, a1, a2, a3, b0, b1, b2, b3, cin, invb	s0, s1, s2, s3, cout	4 bitová sčítačka	invb – negace bitů b
MUL8	a0, a1, a2, a3, b0, b1, b2, b3	s0, s1, s2, s3, s4, s5, s6, s7	Násobička	Osmibitový výsledek
COMPARELEQ	a0, a1, a2, a3, b0, b1, b2, b3	leq, leqn	Komparátor	Porovnání a <= b



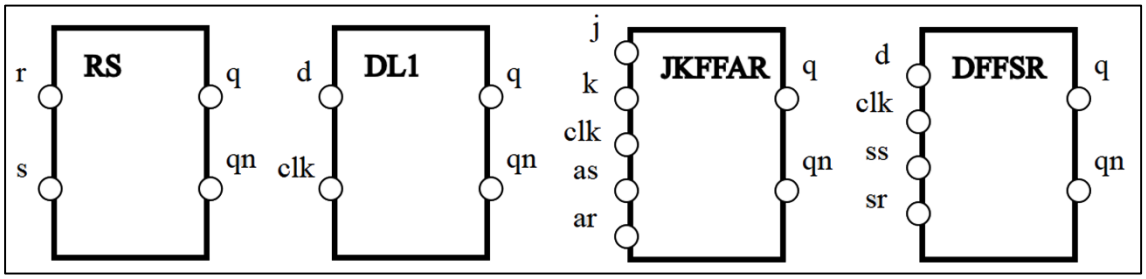
Obrázek 12:Výběr z aritmetických členů

3.2.5 Základní sekvenční členy

Níže popsané členy mají vnitřní paměť, tudíž mohou na stejné vstupy reagovat pokaždé jinak, čímž se liší od výše zmíněných prvků.

Tabulka 5: Seznam základních sekvenčních členů

Název	Vstupy	Výstupy	Popis	Poznámka
RS	r, s	q, qn	RS klopný obvod	r- reset, s - set
DL1	d, clk	q, qn	D klopný obvod	Transparentní, synchronní, clk – hodinový signál
DL1AR	d, clk, ar, as	q, qn	D klopný obvod	Transparentní, asynchronní reset
JKFF	j, k, clk	q, qn	JK klopný obvod	Hranově řízený
JKFFAR	j, k, clk, as, ar	q, qn	JK klopný obvod	Hranově řízený, asynchronní set a reset
JKFFSR	j, k, clk, ss, sr	q, qn	JK klopný obvod	Hranově řízený, synchronní set a reset
DFF	d, clk	q, qn	D klopný obvod	Hranově řízený
DFFAR	d, clk, as, ar	q, qn	D klopný obvod	Hranově řízený, asynchronní set a reset
DFFSR	d, clk, ss, sr	q, qn	D klopný obvod	Hranově řízený, synchronní set a reset



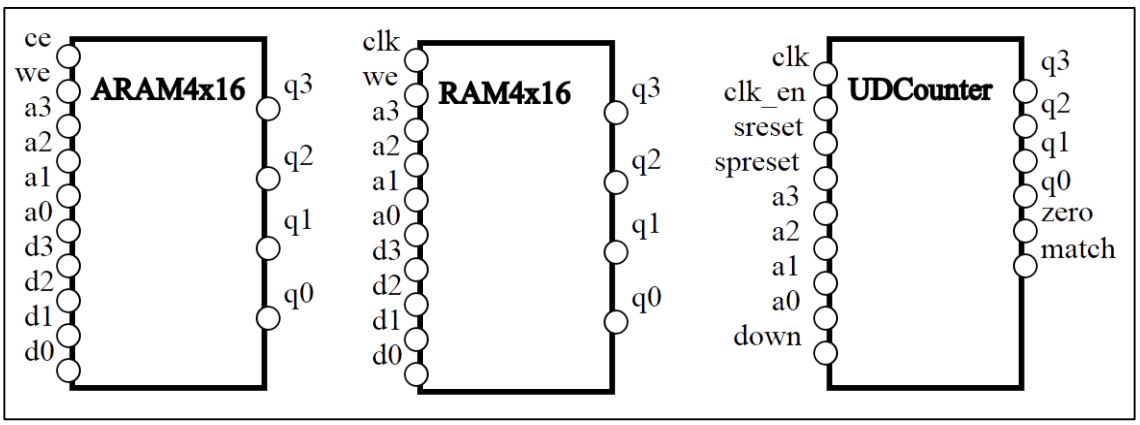
Obrázek 13: Výběr základních sekvenčních členů

3.2.6 Komplexní sekvenční členy

Komplexními sekvenčními členy jsou v tomto případě obvody typu paměť nebo čítač mající složitější vnitřní konstrukci oproti předešlým základním prvkům. V posledním sloupci tabulky jsou popsány vstupy a výstupy. Vynechán je popis výstupů „**q**.“

Tabulka 6: Seznam komplexních sekvenčních členů

Název	Vstupy	Výstupy	Popis	Poznámka
UPDOWNCOUNTER	clk, clk_en, sreset, spreset, a0, a1, a2, a3, down	q0, q1, q2, q3, zero, match	Čtyřbitový čítač	clk – hodinový signál, clk_en – povolení hod. sig., down- čítání dolů, sreset – nulování výstupu, spreset – nastavení poč. hodnot, zero – nulový výstup, match – shoda vstupu a výstupu
ARAM1x16	ce, we, a0, a1, a2, a3, d	q	Jednobitová asynchronní paměť RAM	ce - povolení čipu, we – povolení zápisu, a – čtyřbitová adresa, d - hodnota
ARAM4x16	ce, we, a0, a1, a2, a3, d0, d1, d2, d3	q0, q1, q2, q3	Čtyřbitová asynchronní paměť RAM	ce - povolení čipu, we – povolení zápisu, a – čtyřbitová adresa, d – čtyřbitová hodnota
ARAM4x256	ce, we, a0, a1, a2, a3, a4, a5, a6, a7, d0, d1, d2, d3	q0, q1, q2, q3	Čtyřbitová asynchronní paměť RAM	ce - povolení čipu, we – povolení zápisu, a – osmibitová adresa, d – čtyřbitová hodnota
RAM1x16	clk, we, a0, a1, a2, a3, d	q	synchronní paměť RAM	clk – hodinový signál, we – povolení zápisu, a – čtyřbitová adresa, d - hodnota
RAM4x16	clk, we, a0, a1, a2, a3, d0, d1, d2, d3	q0, q1, q2, q3	Čtyřbitová synchronní paměť RAM	clk – hodinový signál, we – povolení zápisu, a – čtyřbitová adresa, d – čtyřbitová hodnota
RAM4x256	clk, we, a0, a1, a2, a3, a4, a5, a6, a7, d0, d1, d2, d3	q0, q1, q2, q3	Čtyřbitová synchronní paměť RAM	clk – hodinový signál, we – povolení zápisu, a – osmibitová adresa, d – čtyřbitová hodnota
DPRAM4x256	aclk, awe, bclk, bwe, a0, a1, a2, a3, a4, a5, a6, a7, b0, b1, b2, b3, b4, b5, b6, b7, da0, da1, da2, da3, db0, db1, db2, db3	qa0, qa1, qa2, qa3, qb0, qb1, qb2, qb3	Čtyřbitová synchronní dvouportová paměť RAM	aclk, bclk – hodinové signály, awe, bwe – povolení zápisů, a, b – osmibitové adresy, da, db – čtyřbitové hodnoty



Obrázek 14:Výběr komplexních sekvenčních členů

3.3 Popis jednotlivých funkcí v souboru main.js

Webová aplikace obsahuje soubor, v němž jsou naprogramovány všechny funkce psané v jazyce JavaScript. V následujících podkapitolách budou rozebrány.

3.3.1 Deklarace plátna pro vykreslování

První funkcí, kterou zde popíši, je nadefinování plátna (viz Zdrojový kód 1), do něhož se bude následně vše vykreslovat. Dle návodu z internetových stránek autorů knihovny je nejprve nutné nadefinovat element „**Graph**," jenž udává typ použitého vykreslování.

```
var paper = new joint.dia.Paper({
    el: $('#myholder'),
    width: sirka,
    height: vyska,
    gridSize: 1,
    snapLinks: true,
    defaultLink: new joint.shapes.mylib.Vodic,
    model: graph,
    validateConnection: function(vs, ms, vt, mt, e, vl){…}
});
```

Zdrojový kód 1: Definice plátna

Poté se již vytvoří proměnná „**paper**," do níž se dodefinují další potřebné atributy. Dle obrázku je to atribut „**el**," jemuž se přiřadí odkaz na HTML prvek, který má identifikátor „**moje_platno**." Tímto se na zvoleném místě v dokumentu vytvořilo plátno, jemuž dále nastavuji šířku a výšku. Hodnoty jsou upravovány dle rozlišení monitoru, na němž se mají zobrazit. „**GridSize**" udává velikost mřížky plátna, přičemž nastavená hodnota zajišťuje plynulý posun vložených prvků po celé jeho ploše. Při vyšších hodnotách by byl posun skokový. Atribut „**snapLinks**" určuje, zda se mají vodiče v blízkosti portů přichytávat. „**DefaultLink**" udává výchozí spojovací prvek, v mém případě je to definice vodiče z přepracované knihovny. Model nastavuje nadefinovaný typ pro vykreslování. Poslední atribut, „**validateConnection**," pomocí přiřazené funkce průběžně kontroluje spojování jednotlivých portů ve schématu.

3.3.2 Přidávání prvků na plátno

Další důležitou částí je funkce „**přidej“** (viz Zdrojový kód 2), jenž přidává do projektu jednotlivé logické prvky. Do proměnné „**a**“ se načte vybraný prvek z rozevíracího seznamu.

```
function pridej(){
    var a = document.getElementById("prvky").value;
    var b = "new joint.shapes.mylib." +
             a + "({position:{x:140,y:67}});";
    var x = eval(b);
    if(a == 'INPUT'){
        x.attr('.label/text', 'X'+poc_input);
        a = 'X'+poc_input;
        io.push(a.toLowerCase());
        poc_input++;
        graph.addCell(x);
        info_io.push(graph.getCell(x).id);
    }
    else if(a == 'OUTPUT'){…}
    else if(a == 'CLK'){…}
    else{…}
};
```

Zdrojový kód 2: Funkce přidej

Následně je zařazen do připraveného zápisu pro nadefinování prvků z knihovny a uložen do proměnné „**b**“. Takto utvořený příkaz se provede pomocí příkazu „**eval**“ a výsledek uloží do proměnné „**x**.“ Je zde i několik podmínek pro rozlišení prvků, neboť vstupní a výstupní logické členy, včetně hodinového signálu, ukládám do jednoho pole a ostatní prvky do druhého. Uvnitř podmínky pro vstup nastavuji popis prvku včetně očíslování. Rovněž je zde přidání prvku do plátna pomocí funkce „**graph.addCell**.“ Společně s těmito informacemi se ukládají do pole informace s identifikátorem a názvem přidaného prvku. Obdobně je to řešeno v ostatních podmínkách.

Pomocí výše zmíněné funkce je možné do plátna přidávat logické prvky číslicových obvodů. Tyto jsou umístěny na pozici deklarovanou v proměnné „**b**“, a tou je 140 pixelů na ose x a 67 pixelů na ose y. Funkce přidá vždy pouze prvek, který je vybraný v rozevíracím seznamu a toto přidání je aktivováno po stisknutí tlačítka „**přidej“**. Poté si jej uživatel přesune na jakékoliv místo na plátně. Výjimkou je prvek výstup, ten k sobě vykreslí ještě prvek „**BUF**,“ což je opakovač, jenž má jeden vstup a jeden výstup. Přidává se z důvodu ošetření použití výstupu znovu na vstup.

3.3.3 Metody pro exportování

K tomu, aby bylo možné exportovat navržené schéma do externího souboru, je potřeba několik funkcí. Tou hlavní je „**obsah_sem**," ze které jsou spouštěny ostatní funkce, pomocí nichž se pole naplňují potřebnými informacemi. Strukturu výstupního souboru upřesním v kapitole 3.5. Existuje však ještě funkce „**napln_signaly**" (viz Zdrojový kód 3), jež vyhledá v plátně všechny vodiče a uloží je do proměnné „**c**."

```
function napln_signaly(){
    var c = graph.getLinks();
    var signaly = new Array(c.length);
    for(var k = 0; k < c.length; k++){
        signaly[k] = new Array(5);
    }
    for(var i=0 ;i < c.length ; i++){
        signaly[i][0] = "s" + i;
        signaly[i][1] = c[i].get('zdroj').id;
        signaly[i][2] = c[i].get('cil').id;
        signaly[i][3] = c[i].get('zdroj').port;
        signaly[i][4] = c[i].get('cil').port;
    }
    return signaly;
}
```

Zdrojový kód 3: Funkce naplň_signály

Poté si vytvořím dvojrozměrné pole, které má jeden rozměr podle počtu nalezených vodičů a druhý rozměr má 5. V následujícím „**for**" cyklu plním vytvořené pole informacemi o vodičích. V nultém sloupci je název vodiče s pořadím, v jakém byl přidán do plátna. V prvním a druhém sloupci ukládám identifikátor počátečního a koncového uzlu. Do třetího a čtvrtého sloupce vkládám název počátečního a cílového portu. V rámci funkce „**obsah_sem**" ještě zjišťuji, jsou-li vyplněna pole s názvem entity a architektury. Pokud ano, údaje se načtou do připravených proměnných a pokud ne, přednastavil jsem výchozí hodnoty, kterými jsou „**test_circuit**" pro název entity a „**RTL**" pro název architektury.

3.3.4 Funkce vsechno_info

```
function vsechno_info(){
    var cislo = 0;
    var hradla_info = new Array(hradla.length);
    for(var i=0; i < hradla.length;i++){
        hradla_info[i] = new Array(2);
        hradla_info[i][0] = "_";
        hradla_info[i][1] = "_";
    }
    for(var j=0; j < hradla.length;j++){
        for(var k=0; k < hradla_info.length;k++){
            var tempe = hradla_info[k][1].split("_");
            if(tempe[0] == hradla[j]){cislo++;}
        }
        hradla_info[j][0] = info[j];
        hradla_info[j][1] = hradla[j]+"_"+cislo;
        cislo=0;
    }
    return hradla_info;
};
```

Zdrojový kód 4: Funkce všechno_info

V práci rovněž využívám funkci s názvem „**vsechno_info**“ (viz Zdrojový kód 4). Ta prochází všechna hradla v plátně a získává jejich názvy, z nichž následně sestaví dvojrozměrné pole, které obsahuje v nultém sloupci identifikátor hradla a v druhém sloupci název s číslem představujícím, kolikáté je to hradlo stejného druhu.

3.3.5 Obsluha načtení textu ze souboru

Webová aplikace dovoluje nahrát VHDL soubor a vykreslit do plátna v něm uložené schéma. Tato funkce lze provést díky podpoře rozhraní práce se soubory a funkci „**táhni a pusť**“ v HTML 5. Na stránce se nachází prostor, do něhož může uživatel přetáhnout soubor, jenž chce nahrát a zobrazit v plátně. Tento prostor je obsluhován funkcí „**vyber_soubor**“ (viz Zdrojový kód 5). Nejprve jsem nadefinoval tuto metodu a poté jsem prvku stránky, který má tuto funkci použít, přiřadil obsluhu události. Když uživatel přetáhne soubor do vyhrazeného pole, zmíněná metoda načte obsah souboru a uloží jej do proměnné „**nacteny_text**.“ Importovaný soubor musí mít ke svému správnému načtení danou strukturu, a proto jsem před spuštění funkce importu vložil kontrolní podmínku, pomocí níž odfiltruji převážnou část nežádoucích souborů.

Kontroluji načtený text na přítomnost textu „**library IEEE**,“ který by se v každém VHDL souboru měl vyskytovat.

```
function vyber_soubor(evt) {
    evt.stopPropagation();
    evt.preventDefault();
    var soubory = evt.dataTransfer.files;
    var cti = new FileReader();
    cti.onload = function(event) {
        nacteny_text = event.target.result;
        var text = nacteny_text.split(";");
        if(text[0] == "library IEEE"){
            nahrej(nacteny_text);
            alert("Soubor úspěšně načten!");
        }else{alert("Vložen špatný soubor");}}
    cti.readAsText(soubory[0],"UTF-8");
}
```

Zdrojový kód 5: Funkce vyber_soubor

3.3.6 Funkce pro generování testbenche

Mezi další funkce patří generování testovacího souboru testbench. Obsah pro vytvoření souboru je obsluhován funkcí „**tb_sem**“ (viz Zdrojový kód 6).

```
function tb_sem(){
    …
    soubor = fileLoader('tb_vzor.vhd');
    soubor = soubor.split("#");
    soubor[1]= c_jmeno_en;
    soubor[3]= cislo;
    soubor[5]= c_jmeno_en;
    soubor[7]= c_jmeno_arch;
    soubor[9]= c_jmeno_en;
    soubor[11]= c_jmeno_en;
    soubor[13]= vstupy_vystupy;
    soubor[15]= c_jmeno_en;
    if(clk_en == false){
        soubor[17]= "";
        soubor[26]= "";
    }
    soubor[19]= aa.join("");
    soubor[21]= c_jmeno_en;
    soubor[23]= bb.join(",");
    soubor[25]= cc.join("");
    soubor[28]= c_jmeno_arch;
    return soubor.join("");
    …
};
```

Zdrojový kód 6: Funkce generování testbenche

Strukturu výstupního souboru načítám z připraveného souboru „**tb_vzor.vhd**.“ Tento je po načtení rozdělen pomocí dělícího znaku „#“ na části, jež jsou uloženy do pole

„**soubor**.“ V něm jsou uložena textová data, jež jsou pro všechny výstupní soubory společná a rovněž následující místa, na něž vkládám nová data:

a) Pozice 1, 5, 9, 11, 15 a 21 - vložení názvu entity
b) Pozice 7 a 28 - vložení názvu architektury
c) Pozice 3 – vložení čísla, určujícího počet vstupů bez hodinového signálu
d) Pozice 13 – vložení vstupů a výstupů
e) Pozice 17 a 26 – v případě přítomnosti hodinového signálu se data nechají tak, jak jsou, jinak se musí smazat.
f) Pozice 19 – vložení všech použitých signálů z navrženého obvodu
g) Pozice 23 – vložení přiřazení všech portů
h) Pozice 25 – vložení přiřazení vstupních testovacích vektorů

Výsledný text se z jednotlivých částí pole složí do jednoho celku pomocí funkce „**join**“ a následně je vygenerován testovací soubor „**testbench.vhd**.“ Soubor je vzápětí odeslán uživateli ke stažení do počítače.

3.3.7 Funkce nahrání

Předposlední funkcí, kterou zde rozeberu, je importování schématu. Nahrání je realizováno funkcí „**nahrej**,“ kde je parametrem předán načtený text z vloženého souboru. Ihned po zavolání importu je nutné vyčistit paměť všech proměnných a plátna, aby nedošlo k nějaké kolizi. Dále se pomocí několika vložených funkcí naplní proměnné daty z rozčleněného textu. Data jsou představována informacemi o hradlech, vodičích, vstupech, výstupech, identifikátorech hradel a portech. Naplněné proměnné jsou předány metodě „**nahrane_vloz**,“ jež je velmi podobná funkci pro přidávání prvků v úvodu této kapitoly. Navíc je zde přidáno umisťování prvků do plátna.

Exportované soubory a soubory pro nahrání neobsahují informace o poloze jednotlivých prvků, a proto jsem zvolil pro rozmístění prvků vlastní postup. Nejprve nechám vykreslit všechny vstupy pod sebe k levému okraji plátna, poté vykreslím všechny výstupy pod sebe k pravému okraji plátna a následně postupně vykreslím hradla. Kolik jich vložím pod sebe, udává číslo, které se po každém přidaném prvku zvýší o jedna. Když číslo dosáhne určité hodnoty, v mém případě jsem zvolil číslo tři, tak se hradla opět začnou vykreslovat shora a posunuté o několik desítek pixelů doprava tak, aby se nepřekrývala. Po umístění posledního hradla se spustí spojování portů vodiči.

V jednom z naplněných polí jsou informace o přiřazení mezi porty. Tímto polem procházím jednotlivá hradla a spojuji mezi sebou příslušné porty. Každý port má název a každý prvek na plátně má svůj identifikátor. Procházením proměnných pak vyhledávám dle identifikátorů a portů prvky, jež mají být spojeny, a při shodě dané prvky spojím vodičem, který následně přidám do příslušného pole.

3.3.8 Načítání logických prvků

Je zde také funkce pro naplnění rozevíracího seznamu prvků (viz Zdrojový kód 7). Do proměnné „**prvky_vse**“ je uložen načtený text ze souboru „**seznam_prvku**.“

```
function napln_promenne(){
    prvky_vse = fileLoader('seznam_prvku.txt');
    casti = prvky_vse.split("-");
    pole_draty = casti[0].split(";");
    pole_zakl_komb = casti[1].split(";");
    pole_roz_komb = casti[2].split(";");
    pole_arit = casti[3].split(";");
    pole_zakl_sekv = casti[4].split(";");
    pole_roz_sekv = casti[5].split(";");
zmen_seznam();
};
```

Zdrojový kód 7: Naplnění seznamu prvků

Poté je načtený text rozdělen na úseky, kde dělícím znakem je pomlčka. Tyto úseky jsou rozděleny pomocí středníku a ukládány do polí. Pole budou mít následující obsah:

a) „**Pole_draty**“ – vstupy a výstupy (INPUT, OUTPUT, VCC, …)
b) „**Pole_zakl_komb**“ – základní kombinační členy (AND, OR, NOR, …)
c) „**Pole_roz_komb**“ – komplexní kombinační členy (MUX2, DEC18, …)
d) „**Pole_arit**“ – aritmetické členy (HALF-ADDER, MUL8, …)
e) „**Pole_zakl_sekv**“ – základní sekvenční členy (RS, DL1, JKFF, DFF, …)
f) „**Pole_roz_sekv**“ – komplexní sekvenční členy (RAM4x16, ARAM1x16 …)

Po přiřazení dat do všech proměnných se spustí další funkce „**zmen_seznam**.“

3.3.9 Funkce zmen_seznam

```
function zmen_seznam(){
    smaz_seznam();
    var vybranaHodnota;
    var tlacitka = document.getElementsByName("gr1");
    for(var i = 0; i < tlacitka.length; i++) {
      if(tlacitka[i].checked){
      vybranaHodnota = tlacitka[i].value;}}
    switch (vybranaHodnota){…}
    for(var i = 0; i < docasny.length; i++){
        var seznam = document.getElementById("prvky");
        var volba = docasny[i];
        var element = document.createElement("option");
        element.textContent = volba;
        element.value = volba;
        seznam.appendChild(element);
}};
```

Zdrojový kód 8: Funkce změň_seznam

Tato funkce nejprve zavolá podfunkci „**smaz_seznam**“ (viz Zdrojový kód 8), jež projde seznam, a pokud je již naplněn nějakými daty, tak je všechna odstraní. Poté do proměnné „**vybranaHodnota**“ uloží hodnotu vybraného zaškrtávacího políčka. Následuje příkaz „**switch**,“ který zvolí další krok podle vybrané hodnoty v proměnné „**vybranaHodnota**.“ Tím krokem je naplnit proměnnou „**docasny**“ příslušným polem hodnot, jímž je v dalším kroku naplněn rozevírací seznam HTML stránky. Toho je docíleno tak, že se prochází pole „**docasny**“ po jednom prvku a pro každý prvek je vytvořen element „**volba**,“ jemuž je přiřazena textová hodnota z pole „**docasny**“ a element je následně vložen do seznamu.

```
paper.on('cell:pointerdblclick', function(cellView, evt, x, y) {
        cellView.model.remove();
        var aa = info.indexOf(cellView.model.id);
        if(aa != -1){
            info.splice(aa,1);
            hradla.splice(aa,1);
        }
        var bb = info_vstupy.indexOf(cellView.model.id);
        if(bb != -1){
            vstupy.splice(bb,1);
            info_vstupy.splice(bb,1);
        }
});
```

Zdrojový kód 9: Funkce pro mazání prvků

3.3.10 Funkce pro mazání prvků

Poslední funkcí je mazání prvků (viz Zdrojový kód 9). Po dvojkliku na libovolný prvek na plátně se zjistí jeho identifikátor, dle něhož je vyhledán jeho výskyt v poli. Zjistí se pozice v poli a následně je prvek z příslušných polí smazán.

3.4 Struktura knihovny joint.shapes.mylib.js

Pro potřeby diplomové práce bylo po delším zkoumání nutné přepsat dostupnou JavaScriptovou knihovnu. Nutností bylo nadefinovat základní struktury tak, aby se daly jednoduše využít a později rozšířit o další prvky (viz Zdrojový kód 10).

```
joint.shapes.mylib.Hradlo = joint.shapes.basic.Generic.extend({
    markup: '<g class="rotatable"><g class="scalable"><rect/><text
class="jm"/></g><text class="label"/><circle class="input"/><path
class="wire"/><circle class="output"/></g>',
    defaults: joint.util.deepSupplement({
        type: 'mylib.Hradlo',
        size: { width: 1, height: 1 },
       attrs: {
            '.': { magnet: false },
            rect: {
                  width: 50, height: 70,
                  stroke: 'black'
            },
            circle: {
                  r: 6,
                  stroke: 'black',
            },
            text: {
                  fill: 'black',
                  'pointer-events': 'none'
            },
            '.wire': { ref: 'rect', 'ref-y': .5, stroke: 'black'},
            '.label': { text: 'Model', 'ref-x': .3, 'ref-y': .1 },
            '.jm': { text: 'io'},
       }
    }, joint.shapes.basic.Generic.prototype.defaults),
    operation: function() { return true; }
});
```

Zdrojový kód 10: Struktura hradla v knihovně prvků

Základním prvkem je hradlo, které je rozšířením prvku použitého z knihovny „**joint.js**.“ Hradlu jsem nadefinoval atributy, společné všem dalším prvkům:

a) Obdélník – vykreslení logického členu v podobě obdélníku
b) Kruh – vykreslení jednotlivých portů
c) Text – definice stylu vykreslovaného textu

d) Drát – příprava vodiče pro prvky, které jej budou vyžadovat

e) Popisek - vypíše název hradla uvnitř obdélníku

Takto nadefinované obecné hradlo je později rozšiřováno o potřebné prvky. Jednotlivé prvky lze přidávat do oddílu „**markup**.“ Jsou definovány pomocí HTML tříd.

```
joint.shapes.mylib.Hradlo21 = joint.shapes.mylib.Hradlo.extend({
    markup: '<g class="rotatable"><g
class="scalable"><rect/></g><text class="label"/><circle
class="input"/><circle class="input2"/><circle
class="output"/></g><g><text class="jm"/><text class="jm2"/><text
class="jm3"/></g>',
    defaults: joint.util.deepSupplement({
        type: 'mylib.Hradlo21',
        size: { width: 50, height: 70 },
        attrs: {
            '.input': { ref: 'rect', 'ref-x': -2, 'ref-y': 0.3,
magnet: 'passive', port: 'a' },
            '.input2': { ref: 'rect', 'ref-x': -2, 'ref-y': 0.7,
magnet: 'passive', port: 'b' },
            '.output': { ref: 'rect', 'ref-dx': 2, 'ref-y': 0.5,
magnet: true, port: 'q' },
        }
    }, joint.shapes.mylib.Hradlo.prototype.defaults),
    operation: function() { return true; }
});
```

Zdrojový kód 11: Rozšíření základního hradla

Například pro vytvoření dvouvstupového hradla „**AND**“ je nezbytné rozšířit obecné hradlo o jeden vstup (viz Zdrojový kód 11), čehož se docílí přidáním třídy „**<circle class=“input2“>**“ do oddílu „**markup**.“ Hradlo tak obdrží další vstupní port.

```
joint.shapes.mylib.AND =
joint.shapes.mylib.Hradlo21.extend({
    defaults: joint.util.deepSupplement({
        type: 'mylib.AND',
        attrs: {
            '.label': { text: '&', ref: 'rect',
'ref-x': .3, 'ref-y': .1, stroke: 'black'},
             '.jm': { text: 'a', ref: 'rect',
'ref-dx': -70, 'ref-dy': -70 },
            '.jm2': { text: 'b', ref: 'rect',
'ref-dx': -70, 'ref-dy': -40 },
            '.jm3': { text: 'q', ref: 'rect',
'ref-dx': 10, 'ref-dy': -60 },
        }
    },
joint.shapes.mylib.Hradlo21.prototype.defaults)
});
```

Zdrojový kód 12: Vytvoření hradla AND

Dodefinované hradlo musí odkazovat na původní obecné. V tomto rozšíření je již nutné atributům popisujícím vstupní a výstupní porty nastavit reference, od jakého bodu

se bude počítat pozice. Dále se musí nastavit danému portu pozice souřadnicemi „**ref-x**, **ref-y**, **ref-dx** nebo **ref-dy**.“ Význam souřadnic je možné najít na stránkách knihovny v popisu rozhraní. Dalším atributem se určuje, má-li daný port být magnetický nebo pasivní. Magnetickými se nastavují výstupy a vstupy jsou pasivní.

Uživatel kliknutím na výstupní port táhne vodič, který se na vstupní port přichytává. Touto cestou je zaručeno, že na vstup nelze přivést více vodičů. Posledním krokem ke vzniku zvoleného hradla je vytvoření již samotného hradla „**AND**“ (viz Zdrojový kód 12), jež odkazuje na předem vytvořené rozšířené hradlo. Zde se pouze přiřadí popisek prvku a popisky jednotlivých vstupů a výstupů, což je přídavek pro snadnou orientaci ve schématu. Pro navrhování schémat je důležité vědět, co který vstup a výstup znamená. Výše zmíněným postupem lze přidat jakýkoliv prvek do knihovny. O dalším postupu a přidávání prvků se zmíním v kapitole 3.7.

3.5 Struktura výstupního VHDL souboru

Strukturou výstupního souboru rozumíme sestavení požadovaných částí textu, které musí VHDL soubor obsahovat, včetně doplnění údajů ze schématu. Soubor by měl být v pořádku i po syntaktické stránce.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity test_circuit is
      port (
            x1 : in std_logic;
            x2 : in std_logic;
            z1 : out std_logic
      );
end entity test_circuit;

architecture RTL of test_circuit is
      signal and_0_q : std_logic;
      signal buf_0_q : std_logic;

begin
      and_0 : entity work.and_gate
      port map(a => x1,b => x2,q => and_0_q);

      buf_0 : entity work.buf_gate
      port map(a => and_0_q,q => z1);
```

Zdrojový kód 13: Struktura výstupního souboru

Nyní popíši náležitosti generovaného souboru (viz Zdrojový kód 13). Na prvních dvou řádcích se nachází deklarace použitých knihoven VHDL. Další řádky obsahují definici entity, v níž je její název. Uvnitř entity jsou deklarovány všechny vstupy a výstupy navrženého obvodu jako celku. Poté nalezneme na dalších řádcích definici architektury. Ta má název a přiřazení k entitě. Uvnitř architektury je nejprve deklarace všech signálů nacházejících se v obvodu. Dále jsou zde klausule „**begin** a **end**," mezi nimiž jsou postupně vypsána všechna hradla ze schématu. Každé hradlo má svůj očíslovaný název. Za dvojtečkou je odkaz na entitu z knihovny „**lib.vhd**," kde je deklarace všech hradel. Na druhém řádku u deklarace hradla je tzv. port mapa, což je deklarace přiřazení signálů na vstupy a výstupy. První z této dvojice je port, na který je přiřazen signál zapsaný za symbolem přiřazení. Takto jsou deklarovány všechny vstupy a výstupy každého hradla.

3.6 Struktura souboru pro import

Soubory, jež lze importovat, by měly mít stejnou strukturu jako exportované soubory. Důležité je, aby všechny deklarace měly stejný tvar. Pro deklaraci vstupů a výstupů u entity je nutné dodržet pravidlo, že za posledním deklarovaným prvkem nesmí být středník. Naopak u signálů v architektuře musí být středník za každým z nich. Deklarace hradel musí být rovněž stejná a jejich přiřazení jednotlivých portů musí být odděleno čárkou. Jakmile bude nějaká deklarace provedena jinak, je možné, že se schéma vůbec nenačte. V tom případě je nutné webovou stránku znovu načíst a pokusit se nesrovnalost opravit. Jak jsem uvedl v předchozí kapitole, většina nekompatibilních souborů se odfiltruje již před začátkem importu. Jelikož má ale VHDL více možných způsobů zápisu souborů, je nutné pro práci s touto aplikací vycházet z této písemné zprávy, kde je vše detailně popsáno.

3.7 Možnost rozšíření knihovny prvků

Nyní bych shrnul způsob, jakým lze rozšiřovat aplikaci o další prvky. Prvním krokem je dle kapitoly 3.4 nadefinování rozšíření obecného hradla pro potřeby budoucího prvku. Dále vytvoření samotného nového prvku a přidání popisků vstupů a výstupů. Nově vzniklý prvek musí mít rovněž definovanou strukturu v knihovně „**lib.vhd**" a následně je nutné doplnit do textového souboru „**seznam_prvku**," v němž musí mít

stejný název, který je definován za tečkou v oddílu „**type**.“ V souboru „**seznam_prvku“** jsou všechny prvky rozděleny do několika kategorií oddělených pomlčkou. Těmito kategoriemi jsou:

a) Vstupy a výstupy
b) Základní kombinační členy
c) Komplexní kombinační členy
d) Aritmetické členy
e) Základní sekvenční členy
f) Komplexní sekvenční členy

3.8 Možnosti propojení se standardními simulátory

Simulátory VHDL jsem rozdělil na dvě skupiny, online a offline. Online simulátory nemají příliš velké zastoupení a tudíž je jich méně. Při prohledávání internetu jsem nalezl pouze dva.

3.8.1 Online simulátory

Jedním z online simulátorů, který disponuje cloudovým úložištěm a dokáže výsledky zobrazit jako graf, je Tarang EDA [15]. Jeho zajímavou funkcí je možnost pracovat na projektu ve více lidech současně, přičemž se zobrazuje každé kliknutí myši. Druhým simulátorem je EDA Playground [4]. Tento simulátor je komplexnější, má více funkcí než předchozí zmiňovaný. Oba simulátory vyžadují přihlášení uživatele, jinak nelze simulace spustit. Prostředí má přehledné, vlevo na stránce je panel, kde jsou různá nastavení, například výběr programovacího jazyka, jenž chceme simulovat. Dále jsou uprostřed tři pole, dvě z nich slouží k editaci kódu a třetí je výstup konzole. Nad všemi zmíněnými panely je menší panel, v němž jsou ovládací tlačítka pro spouštění, ukládání a přepínání aplikace.

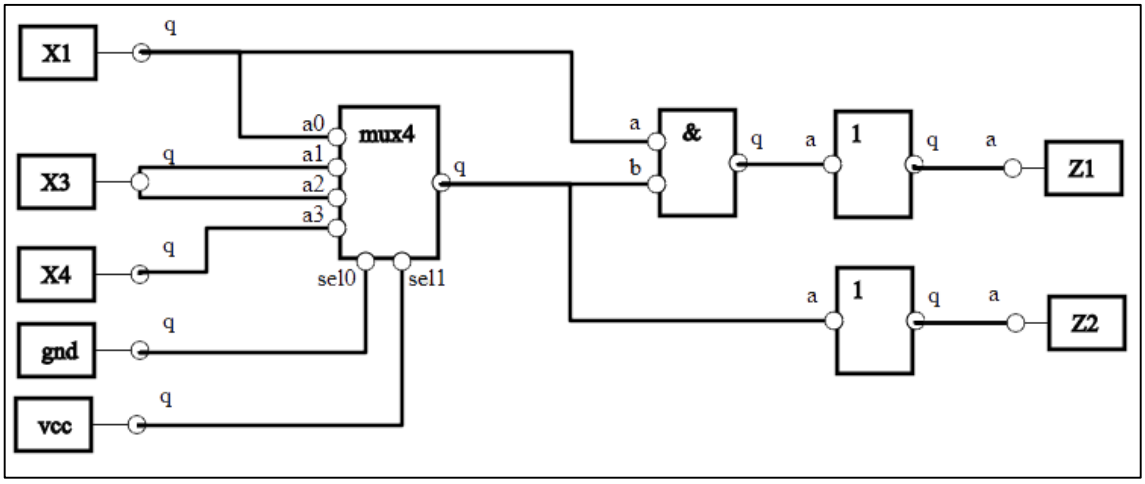
3.8.2 Offline simulátory

Zde uvedu dva simulátory, s nimiž jsem pracoval. Jsou to ISE Simulator od firmy Xilinx a Quartus II Simulator [13] od firmy Altera. Oba podporují VHDL a jsou

nejpoužívanějšími variantami pro výuku. Jejich jediná nevýhoda je, že jsou komerční. Existuje však možnost zkušební verze, která dovolí používat program 30 dní. U programu ISE [1] je dokonce volně ke stažení a používání odlehčená verze nazvaná „**webpack**“ splňující základní požadavek na simulaci a syntézu navržených schémat.

3.9 Ukázka testování a simulace

Ve vytvořené aplikaci jsem pro ukázku navrhl zcela smyšlený obvod (viz Obrázek 15), který má za úkol demonstrovat maximální variabilnost při návrhu schémat.



Obrázek 15: Obvod navržený ve webové aplikaci

Do schématu jsem cvičně umístil pět vstupů „**X1**, **X2**, **X3, GND** a **VCC**.“ Následně jeden multiplexor se čtyřmi vstupy a hradlo „**AND**.“ Poté jsem vložil dva výstupy „**Z1**“ a „**Z2**,“ k nímž se mi automaticky přidaly dva prvky „**BUF**,“ což je odůvodněno výše v textu. Nakonec následovalo pospojování všech prvků pomocí vodičů. Například vstup „**X3**“ jsem spojil se dvěma vstupy multiplexoru, čímž byla dokázána schopnost větvení vodičů. Následujícím krokem bylo stisknutí tlačítka „**export**,“ jež spustilo potřebné metody a sestavilo požadovaný výstupní soubor. Ten byl poté odeslán ke stažení. Dále jsem stiskl tlačítko „**TestBench**,“ jež vzápětí vygenerovalo testovací soubor obvodu, který byl rovněž odeslán ke stažení. Oba soubory jsem vložil do online simulátoru EDA Playground a ten provedl úspěšnou simulaci. Výstupní soubory jsou k nahlédnutí v příloze, včetně úspěšného výsledku simulace.

4 Závěr

Výsledkem diplomové práce je funkční webový editor číslicových obvodů v HTML 5. Je v něm možné navrhovat jednoduchá schémata, která lze následně vyexportovat do VHDL souboru, včetně testovacího souboru testbench. Soubory se schématem lze opět nahrát do aplikace a zobrazit. Aplikace zvládá i větší množství prvků, desítky až stovky na stránce, při zachování plynulosti navrhování. Byla vytvořena knihovna prvků, dále rozšiřitelná o další členy. Na webu je rovněž dostupná ke stažení knihovna, v níž mají jednotlivé prvky definovanou strukturu, jež je důležitá pro simulaci v externích programech. Uživatel si po vygenerování souborů může zvolit cestu, kam chce soubory uložit. Vzhled webu byl vytvořen s důrazem na maximální přehlednost a jednoduchou ovladatelnost.

Neboť se jedná o webovou aplikaci, existují normy pro správnou syntaxi kódu. Kód aplikace jsem nechal zkontrolovat a stránky jsou validní, přičemž syntaxe opravdu odpovídá chystanému standardu HTML 5. Kontrola syntaxe kaskádových stylů je rovněž možná, tudíž jsem ji provedl. Styl použitý v práci je validní a odpovídá kaskádovým stylům verze 3.

Při testování ve zmíněných prohlížečích vše fungovalo správně. Jedinou výjimkou byl prohlížeč Internet Explorer, ve kterém nefungují funkce pro exportování testovacího a výstupního souboru. Vytvořil jsem proto funkci, jež problém částečně řeší. Výstupní soubory lze alespoň stáhnout, avšak ihned poté se celý obsah aplikace smaže. V ostatních prohlížečích žádný problém nenastal. Vzhledem k tomu, že ostatní testované prohlížeče dosahují více než 70 procent počtu uživatelů v ČR, doporučuji používání některého z nich.

Pokud se zamyslím nad budoucím vývojem vytvořené aplikace, možným rozšířením by mohla být optimalizace pro mobilní telefony a tablety, protože se jejich používání stává stále více populárním a většina již teď podporuje některé prvky HTML 5. U mobilních zařízení by však mohl nastat problém při optimalizaci rozlišení, kdy by některé prvky nemusely být na menších displejích dobře viditelné. U menších zařízení je totiž méně místa na navržení rozsáhlejších obvodů. Jiným možným zlepšením by bylo například přidání dalších prvků.

Seznam použitých zdrojů a literatury

[1] ALTERA CORPORATION. *Quartus II Simulator Tools for Education* [online]. 2014 [cit. 2014-05-13]. Dostupné z: http://www.altera.com/education/univ/software/qsim/unv-qsim.html

[2] BARANOVSKIY, Dmitry a Peter ASQUITH. *Raphaël - JavaScript Library* [online]. 2013 [cit. 2014-05-08]. Dostupné z: http://raphaeljs.com/.

[3] CLIENT IO. *JointJS - the HTML 5 JavaScript diagramming library.* [online]. 2009 - 2014 [cit. 2014-05-08]. Dostupné z: http://jointjs.com/.

[4] EDA, Victor. *EDA Playground* [online]. 2013 [cit. 2014-05-13]. Dostupné z: http://www.edaplayground.com/

[5] GOOGLE. *Prohlížeč Chrome* [online]. 2013 [cit. 2014-05-08]. Dostupné z: https://www.google.com/intl/cs/chrome/.

[6] JADRNÝ, Tomáš. *JQuery návod* [online]. 2010 [cit. 2014-05-08]. Dostupné z: http://jquery-navod.cz/.

[7] JANOVSKÝ, Dušan. Úvod do JavaScriptu. JANOVSKÝ, Dušan. *Jak psát web* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://www.jakpsatweb.cz/javascript/javascript-uvod.html.

[8] JQUERY. *JQuery* [online]. 2014 [cit. 2014-05-08]. Dostupné z: https://jquery.org/.

[9] MICROSOFT. *Prohlížeč Internet Explorer* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://windows.microsoft.com/cs-cz/internet-explorer/download-ie.

[10] MOZILLA.ORG. *Prohlížeč Firefox* [online]. 1998 - 2014 [cit. 2014-05-08]. Dostupné z: https://www.mozilla.org/cs/firefox/desktop/.

[11] OPERA SOFTWARE ASA. *Prohlížeč Opera* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://www.opera.com/cs.

[12] PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL*. 1. vyd. Praha: BEN - technická literatura, 2006, 349 s. ISBN 80-730-0198-5.

[13] Quartus II Simulator Tools for Education. ALTERA CORPORATION. *FPGA CPLD and ASIC from Altera* [online]. 2014 [cit. 2014-05-13]. Dostupné z: http://www.altera.com/education/univ/software/qsim/unv-qsim.html

[14] STRATOSIM, LLC. *StratoSim* [online]. 2012 [cit. 2014-05-08]. Dostupné z: https://www.stratosim.com.

[15] TARANG EDA. *Tarang EDA* [online]. 2014 [cit. 2014-05-13]. Dostupné z: http://www.tarangeda.com/main/

[16] W3C. *CSS Namespaces Module Level 3* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://www.w3.org/TR/2014/REC-css-namespaces-3-20140320/.

[17] W3C. *HTML5: W3C Working Draft 25 October 2012* [online]. 2012 [cit. 2014-5- 8]. Dostupné z: http://www.w3.org/TR/2012/WD-html5-20121025/.

[18] WRONEX. *Wronex* [online]. 2013 [cit. 2014-05-08]. Dostupné z: http://www.wronex.com/projects/logic-grid/.

[19] XILINX INC. *ISE WebPACK Design Software* [online]. 2014 [cit. 2014-05-13]. Dostupné z: http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm

[20] YANK, Kevin a Cameron ADAMS. *Začínáme s JavaScriptem*. Vyd. 1. Brno: Zoner Press, 2008, 333 s. ISBN 978-80-86815-94-7.

Příloha A

```
library IEEE;
use IEEE.std_logic_1164.all;
entity test_circuit is
	port (
		x1 : in std_logic;
		x3 : in std_logic;
		x4 : in std_logic;
		z1 : out std_logic;
		z2 : out std_logic
	);
end entity test_circuit;

architecture RTL of test_circuit is
	signal gnd_0_q : std_logic;
	signal vcc_0_q : std_logic;
	signal and_0_q : std_logic;
	signal buf_0_q : std_logic;
	signal buf_1_q : std_logic;
	signal mux4_0_q : std_logic;

begin
	mux4_0 : entity work.mux4
	port map(a0 => x1,a2 => x3,a3 => x4,sel0 => gnd_0_q,sel1 =>
vcc_0_q,q => mux4_0_q,a1 => x3);

	gnd_0 : entity work.gnd
	port map(q => gnd_0_q);

	vcc_0 : entity work.vcc
	port map(q => vcc_0_q);

	and_0 : entity work.and
	port map(a => x1,b => mux4_0_q,q => and_0_q);

	buf_0 : entity work.buf
	port map(a => and_0_q,q => z1);

	buf_1 : entity work.buf
	port map(q => z2,a => mux4_0_q);
```

Obrázek 16: Ukázka výstupního souboru

Příloha B

```
entity testbench_test_circuit is
	generic(C_NUM_INPUTS : integer := 3);
end entity testbench_test_circuit;

architecture RTL of testbench_test_circuit is
	component test_circuit
		port(x1 : in  std_logic;
			x3 : in  std_logic;
			x4 : in  std_logic;
			z1 : out std_logic;
			z2 : out std_logic);
	end component test_circuit;

	constant clk_period : time := 10 ns;

	signal x1 : std_logic;
	signal x3 : std_logic;
	signal x4 : std_logic;
	signal z1 : std_logic;
	signal z2 : std_logic;

	signal input_stimuli : std_logic_vector(C_NUM_INPUTS - 1 downto 0)
:= (others => '0');

begin
	tb_inst : component test_circuit
		port map(x1 => x1, x3 => x3, x4 => x4, z1 => z1, z2 => z2);

	x1 <= input_stimuli(0);
	x3 <= input_stimuli(1);
	x4 <= input_stimuli(2);

	process
	begin
		wait for clk_period;
		for i in 0 to (2 ** C_NUM_INPUTS) - 1 loop
			input_stimuli <= CONV_STD_LOGIC_VECTOR(i,
C_NUM_INPUTS);
			wait for clk_period;
		end loop;
		wait;
	end process;

end architecture RTL;
```

Obrázek 17: Ukázka výstupního souboru testbench

Příloha C

```
-- Compiling entity test_circuit
-- Compiling architecture RTL of test_circuit
-- Loading entity MUX4
-- Loading entity GND
-- Loading entity VCC
-- Loading entity AND
-- Loading entity BUF
Reading /altera-quartus/13.1/modelsim_ase/tcl/vsim/pref.tcl

# 10.1d

Done
```

Obrázek 18: Výstup z webového simulátoru EDA Playground

Příloha D – Struktura přiloženého média

- Zdrojový kód
  - Script – složka se skripty pro aplikaci
- Zpráva
  - diplomova_prace_2014_Jaroslav_Rehak.pdf
  - diplomova_prace_2014_Jaroslav_Rehak.docx
  - originalni_zadani_1.png
  - originalni_zadani_2.png